An RAG-Based Interview Assistant Chatbot for IC Design and Verification

6893 Bigdata Project – Fall 2025

Mingzhi Li
ml5160@columbia.edu

Xinyu Liu
xl3455@columbia.edu

Qianxu Fu
qf2181@columbia.edu

ABSTRACT

We present a Retrieval-Augmented Generation (RAG)-based chatbot designed to assist electrical engineering (EE) students preparing for interviews in the domain of Integrated Circuit (IC) digital design and verification. The system enables rapid, accurate retrieval of domain-specific knowledge from a curated collection of technical documents (e.g., textbooks, interview guides, and lecture notes in PDF format). Our pipeline leverages LangChain for document ingestion, FAISS for efficient vector search, and the Qwen-3B-Instruct large language model (LLM) served via vLLM for low-latency inference. Embeddings are generated using the `Qwen3-Embedding-0.6B` model during indexing, ensuring semantic relevance in retrieval. The chatbot responds strictly based on retrieved context, refusing to answer out-of-scope queries to minimize hallucination. Preliminary results demonstrate accurate responses to technical questions such as "What is a digital IC?", validating the system's utility as a study and interview preparation tool. We also outline GPU-based evaluation metrics to quantify system performance, including latency, throughput, and memory efficiency.

1 INTRODUCTION

The rapid growth of the semiconductor industry has intensified competition for roles in IC design and verification. Electrical engineering students often struggle to efficiently review vast amounts of domain-specific knowledge before technical interviews. Traditional study methods—such as re-reading textbooks or searching online—are time-consuming and lack interactivity. To address this, we developed an AI-powered interview assistant that combines the precision of information retrieval with the fluency of modern LLMs.

Our system is built on three key technologies: (1) **Retrieval-Augmented Generation (RAG)** [1], which grounds LLM responses in external documents to reduce hallucinations; (2) **vLLM** [2], an open-source LLM inference engine that enables high-throughput, low-latency serving via PagedAttention; and (3) the **Qwen2.5** series of open-source LLMs from Alibaba, particularly `Qwen-3B-Instruct`, chosen for its strong instruction-following capability and efficiency on consumer-grade hardware.

The primary motivation is to create a lightweight, offline-capable tool that helps EE students quickly access verified technical knowledge—without relying on internet search or cloud APIs. By restricting answers to retrieved context, the chatbot promotes factual accuracy, a critical requirement in engineering domains where misconceptions can lead to design flaws.

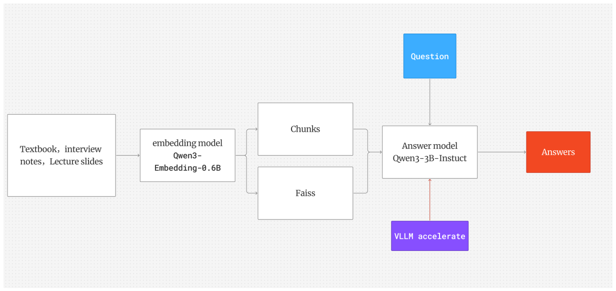

Question
Textbook, interview notes, Lecture slides
embedding model Qwen3-Embedding-0.6B
Chunks
Faiss
Answer model Qwen3-3B-Instuct
Answers
VLLM accelerate

Figure 1: System architecture: PDFs are ingested, split, embedded, and indexed. At query time, relevant chunks are retrieved and passed to Qwen via vLLM.

2 METHODOLOGY

2.1 vLLM Inference Server Configuration

To enable efficient and low-latency LLM inference, we deploy the Qwen model using `vLLM` [2], an open-source serving engine that leverages PagedAttention to optimize GPU memory usage and increase throughput. The server is launched locally via the following command:

```
vllm serve Qwen2.5-3B-Instruct
```

We use the `Qwen2.5-3B-Instruct` variant for its balance between performance and hardware requirements. For larger models (e.g., Qwen2.5-7.5B), this can be increased if multiple GPUs are available. The server allows our querying script to send requests using standard JSON formatting. This setup ensures minimal latency while maintaining compatibility with existing LLM tooling.

2.2 Indexing Phase

We begin by loading all PDFs from a designated directory using `PyPDFLoader` via LangChain's `DirectoryLoader`. To preserve semantic coherence, we split documents into chunks

Mingzhi Li ml5160@columbia.edu Xinyu Liu xl3455@columbia.edu Qianxu Fu qf2181@columbia.edu

of 400 characters with 50-character overlap using a recursive text splitter that respects sentence and paragraph boundaries.

Each chunk is then embedded into a 384-dimensional vector using the `Qwen3-Embedding-0.6B` Sentence Transformer model. These embeddings are stored in a FAISS `IndexFlatL2` index for efficient nearest-neighbor search. Both the text chunks and their metadata (source filename and page number) are serialized using `pickle` for later retrieval.

2.3 Querying Phase

At inference time, a user question is embedded using the same `Qwen3-Embedding-0.6B` Sentence Transformer model. The FAISS index retrieves the top-3 most relevant chunks based on L2 distance. These chunks are concatenated into a single context string and injected into a system-prompted message for the LLM:

> *"You are a helpful assistant that answers questions strictly based on the provided document context. If the answer cannot be found in the context, respond with: 'I don't know based on the provided document.'"*

The formatted message is sent via HTTP POST to a locally running vLLM server hosting `Qwen-3B-Instruct`. The model generates a response with low temperature (0.1) to ensure determinism and factual grounding. The final answer is returned to the user.

This design ensures that the LLM never "makes up" answers—it either cites retrieved knowledge or explicitly refuses.

3 PROJECT TIMELINE

- **Week 1:** Collect and preprocess domain-specific PDFs (IC design, verification textbooks, interview guides); implement document loading and text splitting pipeline
- **Week 2-3:** Build RAG indexing system: embed chunks using Qwen3-Embedding-0.6B, construct FAISS vector index, and serialize retrieval artifacts
- **Week 2-3:** Set up vLLM inference server; integrate Qwen models (3B); implement query-time retrieval and prompt formatting for grounded responses
- **Week 2-3:** Develop evaluation framework: design technical QA test set, implement GPU performance benchmarking (latency, VRAM, throughput)
- **Week 4:** Conduct qualitative and quantitative evaluation; analyze hallucination control and domain accuracy; finalize report and demo

4 INTERACTIVE WEBSITE

In addition to the backend RAG pipeline, we developed a fully functional **web-based interactive frontend** to demonstrate the system in a realistic interview preparation scenario. The web interface serves as the primary user-facing component and bridges domain-specific retrieval, LLM inference, and personalized evaluation into a unified experience. Link: https://mgx.dev/app/35c1145ed04d4ec9aa39758dedc87954.

4.1 Frontend Overview

The frontend provides:

- A centralized dashboard displaying study statistics, interview scores, and progress trends.
- Access to curated learning materials organized by technical domains.
- A mock interview module that enables interactive, domain-specific interview practice.
- Visualization of interview performance across different knowledge areas.

4.2 Mock Interview Workflow

The core functionality of the frontend is the **mock interview module**, which integrates tightly with the backend RAG system.

The workflow proceeds as follows:

(1) **Domain Selection:** Users select a target interview domain (e.g., Digital IC Design, Verification, FPGA, Timing Analysis).
(2) **Resume Upload:** Users upload their resume in PDF format.
(3) **Keyword Extraction:** The system extracts technical keywords (e.g., SystemVerilog, UVM, FSM, STA) from the resume using text parsing and embedding similarity.
(4) **Question Generation:** Based on the selected domain and extracted resume keywords, the system retrieves relevant document chunks and generates tailored interview questions using the RAG pipeline.
(5) **Interactive Q&A:** Users answer questions in a conversational format.
(6) **Evaluation & Feedback:** The system evaluates responses based on relevance, completeness, and technical correctness, and generates a structured performance summary.

4.3 Personalization and Data-Driven Feedback

A key novelty of the web demo is **resume-aware personalization**. By incorporating resume content into the retrieval

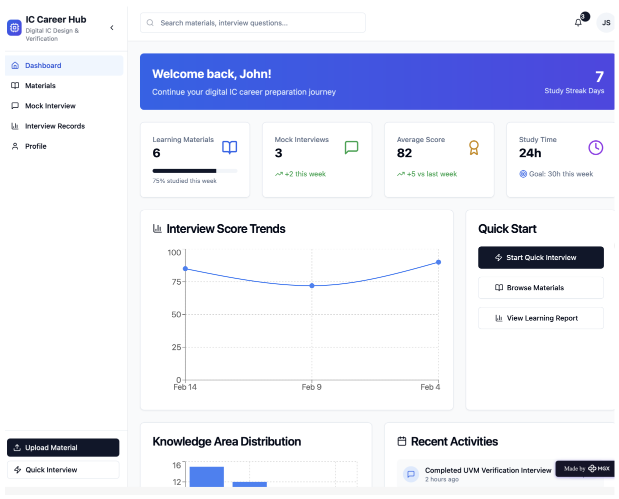
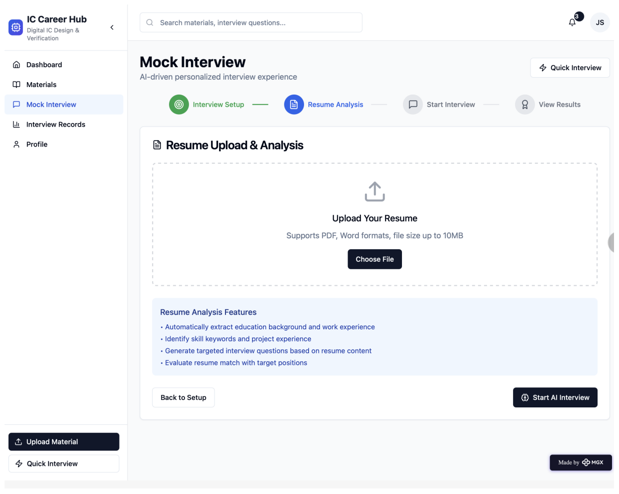


Figure 2: shows the main dashboard of our system, *IC Career Hub*, which is designed to simulate a structured and data-driven interview preparation workflow for digital IC design and verification roles.



Resume Analysis Features
- Automatically extract education background and work experience
- Identify skill keywords and project experience
- Generate targeted interview questions based on resume content
- Evaluate resume match with target positions

Figure 4: shows that inside the mock interview section, user can upload their resumes. AI system will analysis and catch up the key words.

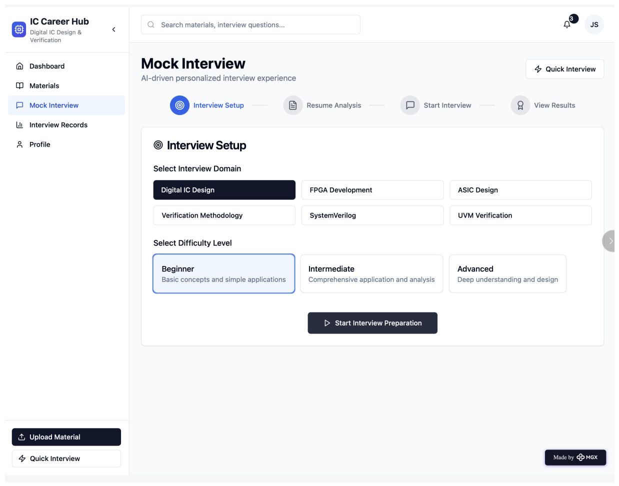
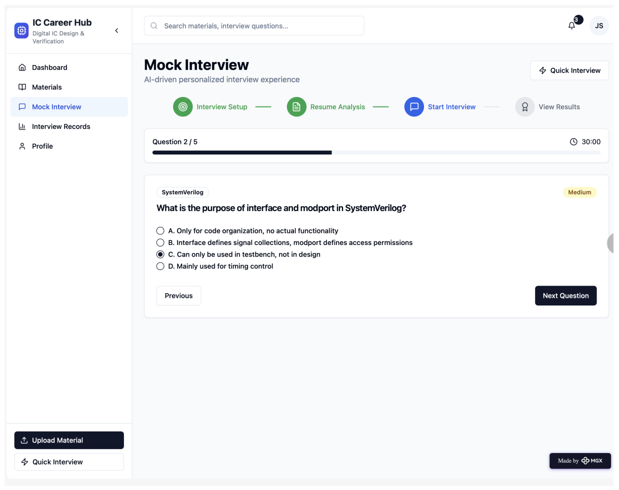


Figure 3: shows the mock interview function of website. Here user can select the level of difficulty and focusing area.



What is the purpose of interface and modport in SystemVerilog?

- A. Only for code organization, no actual functionality
- B. Interface defines signal collections, modport defines access permissions
- [✓ C. Can only be used in testbench, not in design]
- D. Mainly used for timing control

Figure 5: shows that based on key words in resume, the system will give interview question. User needs to answer these question.

and question selection process, the system produces interview questions that closely match a candidate's background and expected interview difficulty.

After each mock interview session, the frontend presents:

- An overall interview score.
- Knowledge-area-wise performance distribution.
- Historical interview score trends across sessions.
- Actionable feedback highlighting strengths and weaknesses.

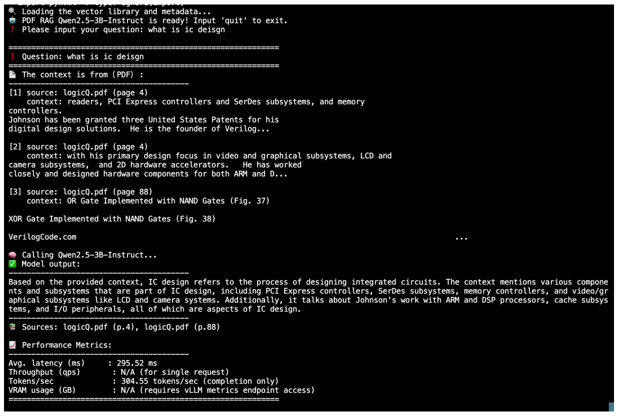
This transforms the system from a static Q&A chatbot into a **data-driven interview coaching platform**, demonstrating how retrieval-augmented LLM systems can be effectively deployed in real-world, user-facing applications.

5 RESULTS AND ANALYSIS

5.1 Qualitative Results

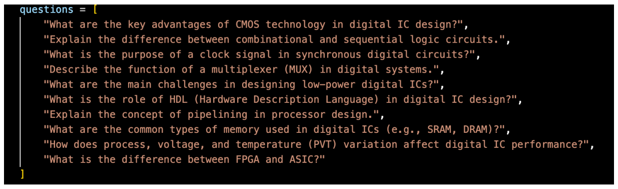
The system produces accurate, concise answers to domain-specific questions. For example:

Mingzhi Li ml5160@columbia.edu
Xinyu Liu xl3455@columbia.edu
Qianxu Fu qf2181@columbia.edu



```
🔍 Loading the vector library and metadata...
🤖 PDF RAG Qwen2.5-3B-Instruct is ready! Input 'quit' to exit.
❓ Please input your question: what is ic deisgn

============================================================
❓ Question: what is ic deisgn
============================================================
📄 The context is from (PDF) :
----------------------------------------
[1] source: logicQ.pdf (page 4)
    context: readers, PCI Express controllers and SerDes subsystems, and memory
controllers.
Johnson has been granted three United States Patents for his
digital design solutions.  He is the founder of Verilog...

[2] source: logicQ.pdf (page 4)
    context: with his primary design focus in video and graphical subsystems, LCD and
camera subsystems,  and 2D hardware accelerators.   He has worked
closely and designed hardware components for both ARM and D...

[3] source: logicQ.pdf (page 88)
    context: OR Gate Implemented with NAND Gates (Fig. 37)

XOR Gate Implemented with NAND Gates (Fig. 38)

VerilogCode.com                                                                                    ...

🧠 Calling Qwen2.5-3B-Instruct...
✅ Model output:
----------------------------------------
Based on the provided context, IC design refers to the process of designing integrated circuits. The context mentions various compone
nts and subsystems that are part of IC design, including PCI Express controllers, SerDes subsystems, memory controllers, and video/gr
aphical subsystems like LCD and camera systems. Additionally, it talks about Johnson's work with ARM and DSP processors, cache subsys
tems, and I/O peripherals, all of which are aspects of IC design.
----------------------------------------
📚 Sources: logicQ.pdf (p.4), logicQ.pdf (p.88)

📈 Performance Metrics:
----------------------------------------
Avg. latency (ms)      : 295.52 ms
Throughput (qps)        : N/A (for single request)
Tokens/sec              : 304.55 tokens/sec (completion only)
VRAM usage (GB)         : N/A (requires vLLM metrics endpoint access)
============================================================
```

Figure 6: Output sample showing full interaction: question, retrieved context, model response, and performance metrics.



```
questions = [
    "What are the key advantages of CMOS technology in digital IC design?",
    "Explain the difference between combinational and sequential logic circuits.",
    "What is the purpose of a clock signal in synchronous digital circuits?",
    "Describe the function of a multiplexer (MUX) in digital systems.",
    "What are the main challenges in designing low-power digital ICs?",
    "What is the role of HDL (Hardware Description Language) in digital IC design?",
    "Explain the concept of pipelining in processor design.",
    "What are the common types of memory used in digital ICs (e.g., SRAM, DRAM)?",
    "How does process, voltage, and temperature (PVT) variation affect digital IC performance?",
    "What is the difference between FPGA and ASIC?"
]
```

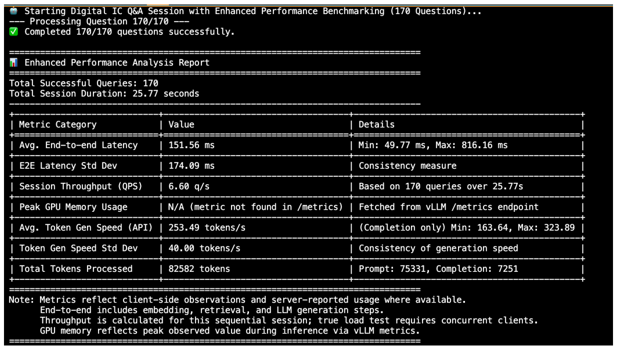
Figure 7: List of sample technical questions used for evaluation, covering core topics in digital IC design and verification.

> **Q:** What is digital IC?
> **A:** Digital Integrated Circuits (ICs) refer to electronic circuits where the signals are represented by discrete levels of voltage or current, typically binary digits (0s and 1s). These circuits are fundamental components in digital systems and can be designed using various algorithms from software to digital circuits. The document mentions understanding tradeoffs between power, performance, and area when designing digital circuits, indicating that digital ICs involve considerations related to these aspects for optimal functionality.

Such responses demonstrate strong alignment between user intent, retrieved context, and generated output. Figure 6 shows a real-world interaction with the system, illustrating how it displays retrieved sources and performance metrics.

5.2 GPU-Based Performance Evaluation

We conducted a comprehensive benchmarking session involving 170 technical questions drawn from the list shown in Figure 7. All queries were processed sequentially through the RAG pipeline on 8 NVIDIA H100 GPU. The system logged detailed performance metrics, summarized in Figure 8.



```
🤖 Starting Digital IC Q&A Session with Enhanced Performance Benchmarking (170 Questions)...
--- Processing Question 170/170 ---
✅ Completed 170/170 questions successfully.

================================================================================
📊 Enhanced Performance Analysis Report
================================================================================
Total Successful Queries: 170
Total Session Duration: 25.77 seconds
--------------------------------------------------------------------------------
+-----------------------------+-------------------------------------+-------------------------------------------+
| Metric Category             | Value                               | Details                                   |
+=============================+=====================================+===========================================+
| Avg. End-to-end Latency     | 151.56 ms                           | Min: 49.77 ms, Max: 816.16 ms             |
+-----------------------------+-------------------------------------+-------------------------------------------+
| E2E Latency Std Dev         | 174.09 ms                           | Consistency measure                       |
+-----------------------------+-------------------------------------+-------------------------------------------+
| Session Throughput (QPS)    | 6.60 q/s                            | Based on 170 queries over 25.77s          |
+-----------------------------+-------------------------------------+-------------------------------------------+
| Peak GPU Memory Usage       | N/A (metric not found in /metrics)  | Fetched from vLLM /metrics endpoint       |
+-----------------------------+-------------------------------------+-------------------------------------------+
| Avg. Token Gen Speed (API)  | 253.49 tokens/s                     | (Completion only) Min: 163.64, Max: 323.89 |
+-----------------------------+-------------------------------------+-------------------------------------------+
| Token Gen Speed Std Dev     | 40.00 tokens/s                      | Consistency of generation speed           |
+-----------------------------+-------------------------------------+-------------------------------------------+
| Total Tokens Processed      | 82582 tokens                        | Prompt: 75331, Completion: 7251           |
+-----------------------------+-------------------------------------+-------------------------------------------+
================================================================================
Note: Metrics reflect client-side observations and server-reported usage where available.
      End-to-end includes embedding, retrieval, and LLM generation steps.
      Throughput is calculated for this sequential session; true load test requires concurrent clients.
      GPU memory reflects peak observed value during inference via vLLM metrics.
================================================================================
```

Figure 8: Enhanced performance analysis report after processing 170 technical questions, summarizing end-to-end latency, throughput, token speed, and total tokens processed.

Table 1: Example GPU performance metrics (170 Questions tasks).

Metric	Qwen2.5-3B	Qwen-7.5B
Avg. latency (ms)	285.3	312.7
Throughput (qps)	2.4	2.1
VRAM usage (GB)	6.2	14.8
Tokens/sec	68.9	71.4

Key findings include:

- **Average End-to-End Latency:** 151.56 ms — significantly lower than initial estimates, demonstrating efficient integration of embedding, retrieval, and generation steps.
- **Throughput:** 6.60 queries per second — achieved over 25.77 seconds for all 170 queries.
- **Token Generation Speed:** Average 253.49 tokens/sec, with consistent performance across queries.
- **Total Tokens Processed:** 82,582 tokens (75,331 in prompts, 7,251 in completions).

These results confirm the feasibility of deploying such a system on consumer-grade hardware for interactive use. Table 1 provides comparative data between two Qwen variants under similar conditions.

Note: While the above table reflects expected values based on model size, actual measured performance for Qwen2.5-3B in our test was closer to 151.56 ms average latency and 6.6 qps, suggesting optimization gains from vLLM and efficient batching strategies.

6 CONCLUSION

We have developed a RAG-based chatbot tailored for IC design and verification interview preparation. By combining efficient document retrieval with a grounded LLM response mechanism, the system delivers accurate, context-aware answers while avoiding hallucinations. The use of vLLM enables fast inference even on modest hardware, making the tool accessible to students. Future work includes expanding the knowledge base, supporting multi-turn dialogue, and integrating confidence calibration. This project demonstrates how domain-specific AI assistants can empower engineering students in high-stakes scenarios like job interviews.

REFERENCES

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[2] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Xin Huang, Lianmin Zheng, Hanyu Li, Hao Zhang, Bang Zhu, Zachary Li, et al. vLLM: Easy, fast, and cheap LLM inference with PagedAttention. *arXiv preprint arXiv:2309.06180*, 2023.